The Halting Problem

Reading

Section 4.2

The Universal Turing Machine

The work we did in the previous section naturally extends to Turing Machines. In fact we can define:

$$A_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM that accepts } w\}$$

So this language captures all pairs of a Turing Machine and a string, such that the Turing Machine accepts the string.

> A_{TM} is Turing-recognizable.

The machine that recognizes this language is called a *Universal Turing Machine*: It is effectively a Turing Machine that simulates the computation of all Turing Machines. It goes as follows:

1. On input a TM M and a string w.
2. It simulates M on input that string w.
3. If M enters its accept or reject states, then our universal Turing machine also enters its accept (resp. reject) state.
4. If M loops (runs forever), so does our universal Turing machine.

Notice that this is most definitely not a decider: It has no way to determine whether M will loop or not, other than actually simulating it and running forever.

The Halting Problem

The Halting Problem effectively boils down to the following:

> A_{TM} is not decidable.

In other words, there can never be a Turing Machine/program that is guaranteed to terminate and that can decide, given an arbitrary Turing Machine/program and its input, whether that program would terminate. In terms of programming languages, this means that we cannot write a program that can tell us if programs in our language would terminate. This has implications in terms of what a type system can do, which go a bit beyond the scope of this course.

This is also our first example of a specific language that is Turing-recognizable but not decidable.

Now let us discuss the proof of this extremely important result. It is based on another assertion:

> Consider the language:
>
> $$\text{NONSELFACC} = \{\langle M \rangle \mid M \text{ does not accept on input } \langle M \rangle\}$$
>
> of all Turing Machines that do not accept when given themselves as input.
>
> Then NONSELFACC is not decidable.

Before we prove this result, let us see why this would imply that A_{TM} is also not decidable.

If A_{TM} were decidable, denote by H a decider. We then construct a decider for NONSELFACC as follows:

- Given a Turing machine M, use H to run M on input its string representation $\langle M \rangle$.
- If H accepts, then we reject.
- If H rejects, then we accept.

This describes a decider for NONSELFACC. But since that was not a decidable language, we have a contradiction.

Now let us discuss why NONSELFACC is not decidable. Suppose instead that it was decidable, and that D was its decider. Then we have a problem:

- If $D \in$ NONSELFACC, then it means that NONSELFACC's decider must have accepted D, and so running D on input $\langle D \rangle$ must have accepted. But since D was a decider for NONSELFACC, this means that $D \notin$ NONSELFACC.
- Conversely, if $D \notin$ NONSELFACC, then this means that running D with input itself, $\langle D \rangle$, would accept. But since D is the decider of NONSELFACC and it just accepted input $\langle D \rangle$, this must mean that $D \in$ NONSELFACC.

So we have a contradiction, as D cannot both be in NONSELFACC and not be in it. So D cannot exist in the first place.

A non-recognizable language

We end this chapter with an example of a language that's not even Turing-recognizable.

> The language that is the complement of A_{TM} is not Turing-recognizable.
>
> In general, the complement of any language that is Turing-recognizable but not decidable, is itself not Turing-recognizable.

This follows simply from the observation that if both a language and its complement were Turing-recognizable, then we could build a decider for the language:

1. On input w:
2. Simultaneously simulate the recognizer Turing Machines for the language and its complement, both on input w.
3. Since w is either in the language or its complement, one of those recognizers will have to accept w. We can then terminate with either accept or reject, depending on which recognizer accepted.

Exercise: What about the following slightly different language:

> Show that the language
>
> $$L = \{\langle M, w\rangle \mid M \text{ is a TM that does not accept on input } w\}$$
>
> is not Turing-recognizable.